Identificação e Geração de Oráculos de *Bad Smells* em Software

Rafael Prates Ferreira Trindade

Orientadora: Mariza Andrade da Silva Bigonha
Coorientadora: Kecia Aline Marques Ferreira

Defesa de Mestrado
Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Ciência da Computação

Agosto de 2020

Roteiro da Apresentação

Introdução

Bad Smells

- Um *bad smell* é um sintoma de que em um determinado trecho de código pode existir um problema mais profundo de projeto que pode comprometer a manutenibilidade do software
- Fowler et al. (1999) definiram um conjunto de de 22 *bad smells*
- Brown et al. (1998) contribuem com a descrição de 14 *bad smells* separados em momentos: na gestão, na arquitetura ou no desenvolvimento

Introdução

Detecção de *Bad Smells*

- Uma quantidade excessiva de *bad smells* em um software dificulta a manutenção e a evolução dele (Sharma andSpinellis, 2018)
- **Abordagem Manual** exige a inspeção do código por uma pessoa experiente que identificará a ocorrência de um *bad smell*
- **Abordagem Automática** é a implementação de técnicas de detecção. Fernandes et al. (2016) identificaram seis técnicas de detecção de *bad smells* em 84 ferramentas

Introdução

Oráculo

- Conjunto de dados considerados verdadeiros acerca de determinada característica de um software
- **Oráculo de *bad smell*** consiste em um arquivo independente de formato que indica quais classes possuem uma ocorrência de um *bad smell*
- Pode-se gerar um oráculo de *bad smells* por meio de (i) inspeção manual de um sistema, (ii) varredura automática ou (iii) combinação de ambas as técnicas

Introdução

Motivação

- Paira uma dúvida sobre a eficácia das ferramentas propostas
- A avaliação de técnicas e ferramentas de detecção de *bad smells* pode ser realizada via comparação dos seus resultados com um oráculo
- Oráculos produzidos via inspeção manual de software são difíceis de serem encontrados e produzidos

Introdução

Objetivos

Identificar, criar e disponibilizar oráculos de *bad smells* para software orientado por objetos.

Objetivos específicos:

1. Realizar uma revisão sistemática da literatura para identificar os oráculos de *bad smells* disponíveis
2. Desenvolver uma ferramenta para auxiliar a inspeção manual de software para identificação de *bad smells*
3. Criar e disponibilizar um oráculo de *bad smells* para software Java

Revisão Sistemática da Literatura

Questões de pesquisa

QP1 - Quais oráculos de bad smells foram propostos na literatura?

QP2 - Em quais linguagens de programação os sistemas que compõem os oráculos são implementados?

QP3 - Qual é o tamanho dos sistemas que compõem os oráculos?

QP4 - Quais bad smells são considerados pelos oráculos?

QP5 - Quais abordagens foram usadas para criar os oráculos?

Revisão Sistemática da Literatura

String de busca

*((“oracle"OR “benchmark") AND ((“anti-pattern"AND “software") OR
(((“bad"OR “design"OR “code"OR “architecture") AND (“smell")) OR
((“design"OR “code") AND “anomaly"))))*

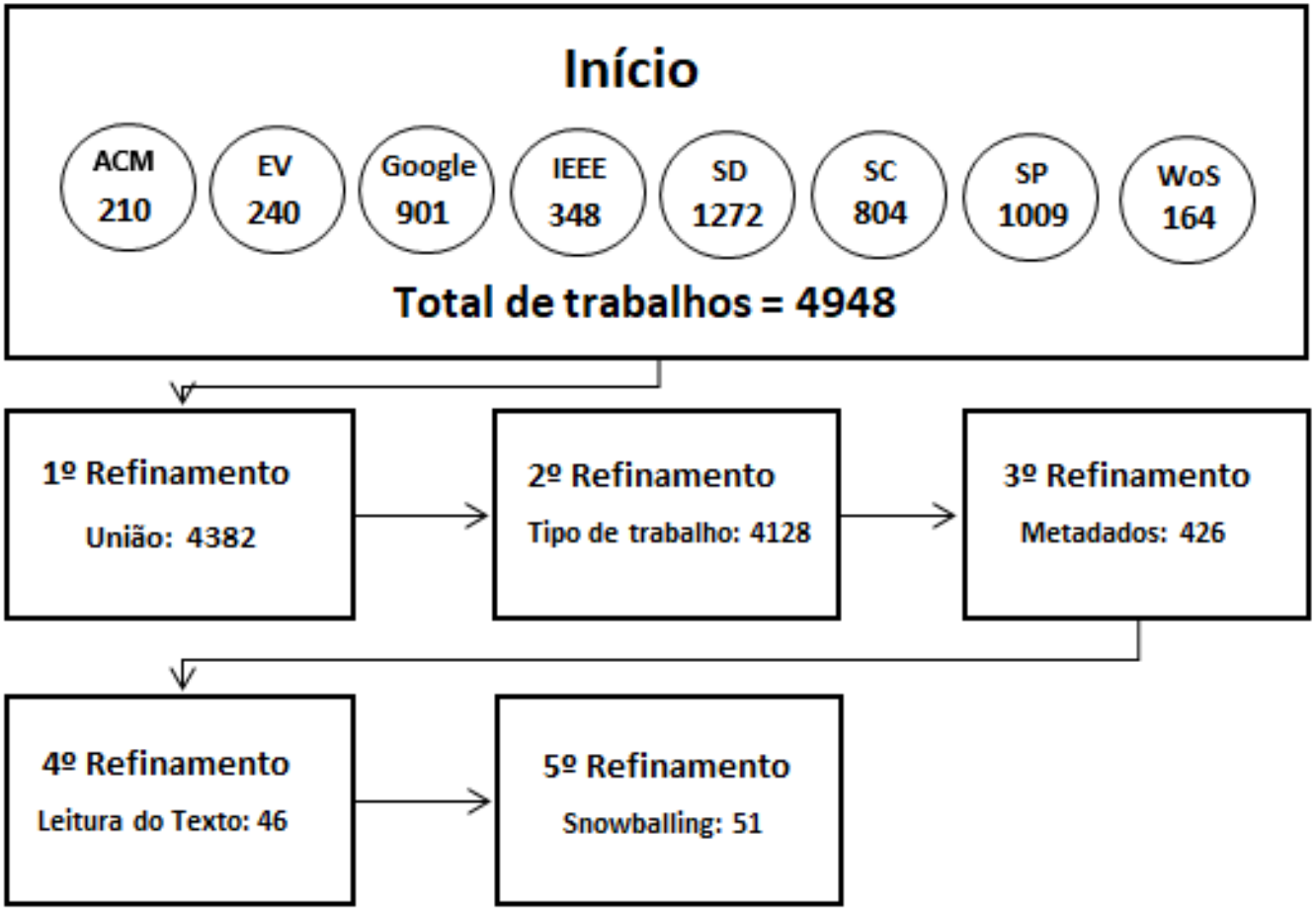
Revisão Sistemática da Literatura

Repositórios Digitais

- ACM Digital Library (http://dl.acm.org/)
- IEEE Xplore (http://ieeexplore.ieee.org/)
- Science Direct (http://www.sciencedirect.com/)
- Scopus (http://www.scopus.com/)
- Springer (https://link.springer.com/)
- Web of Science (http://webofknowledge.com/)
- Engineering Village (http://www.engineeringvillage.com/)
- Google Scholar (https://scholar.google.com.br/)

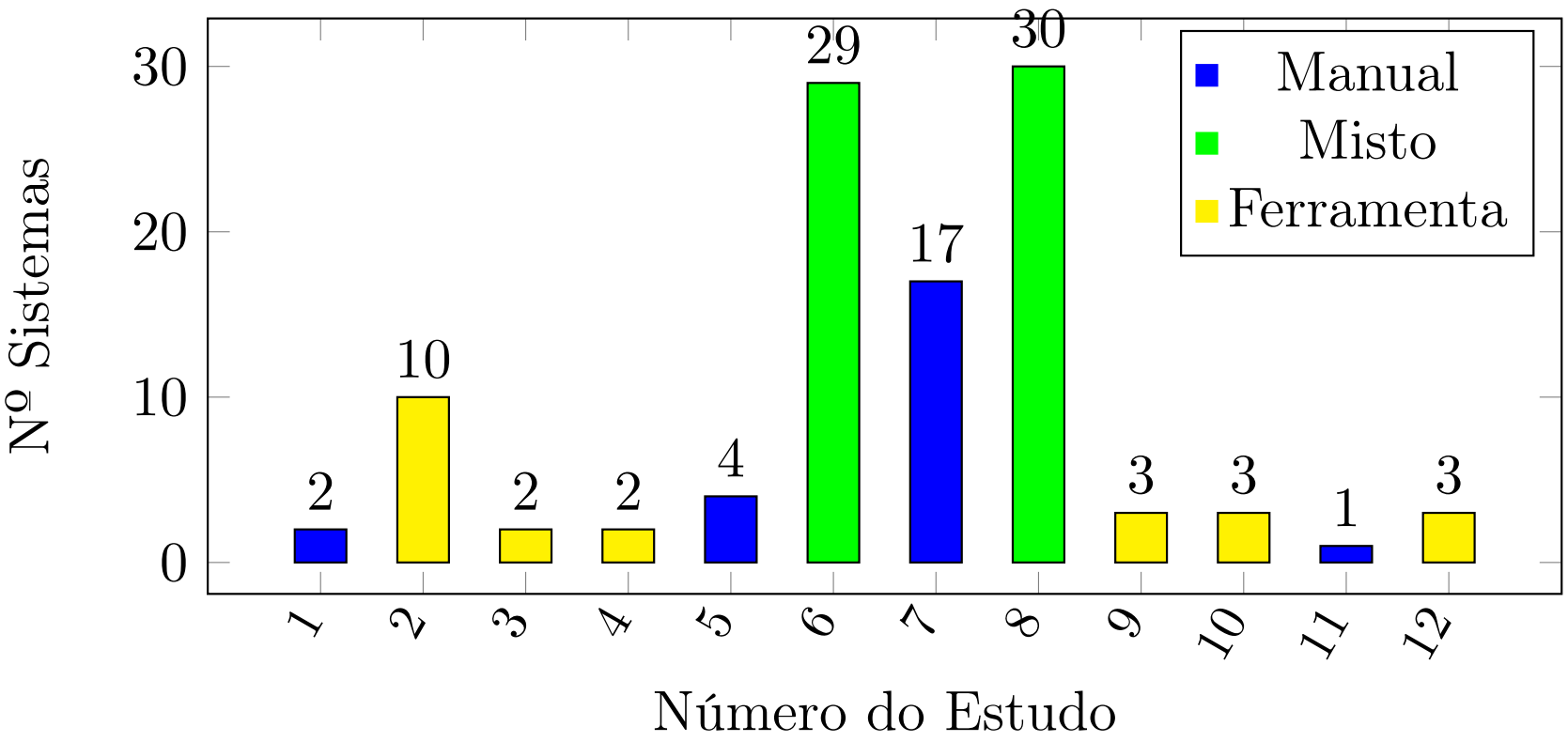
Revisão Sistemática da Literatura

Refinamentos



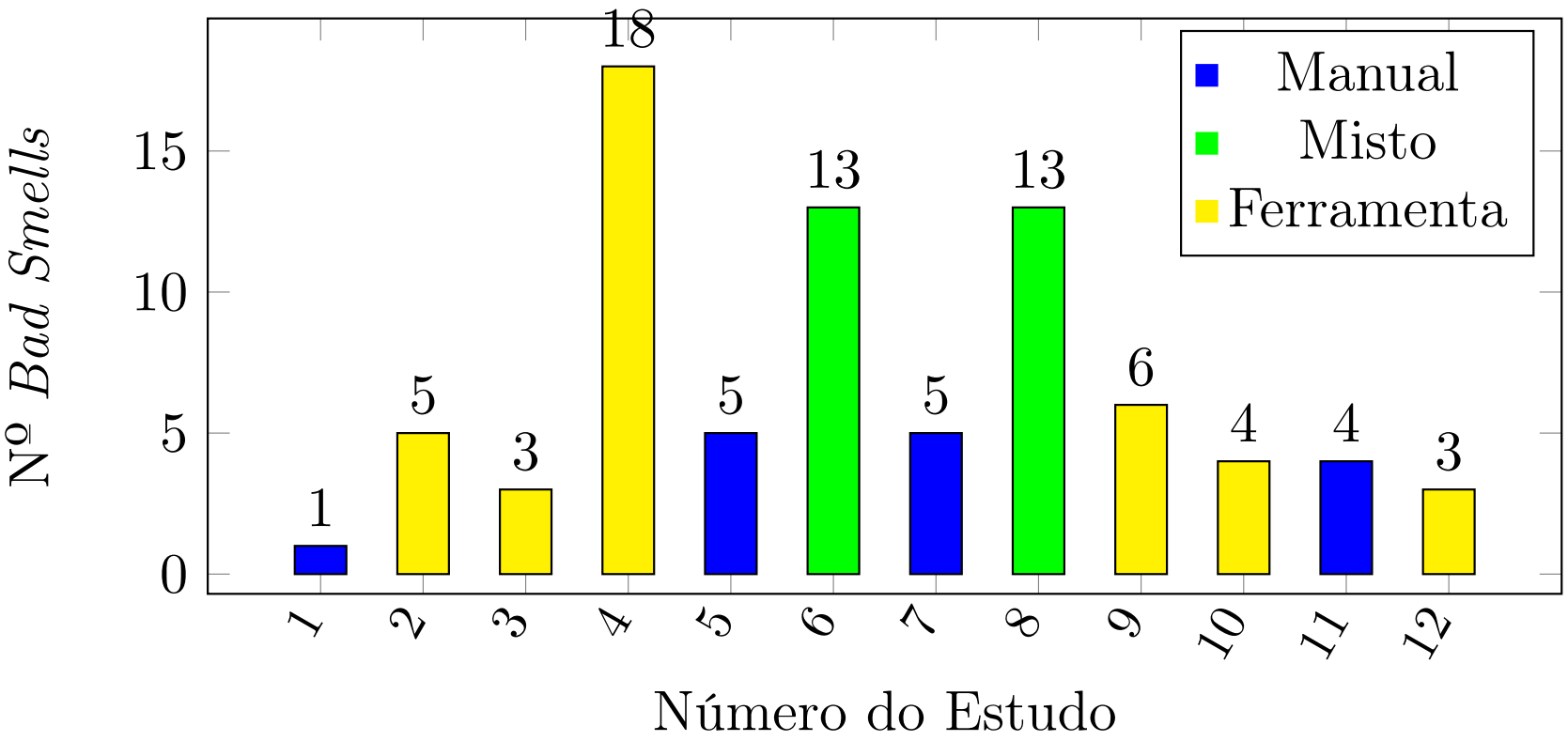
Revisão Sistemática da Literatura

QP1 - Quais oráculos de *bad smells* foram propostos na literatura?



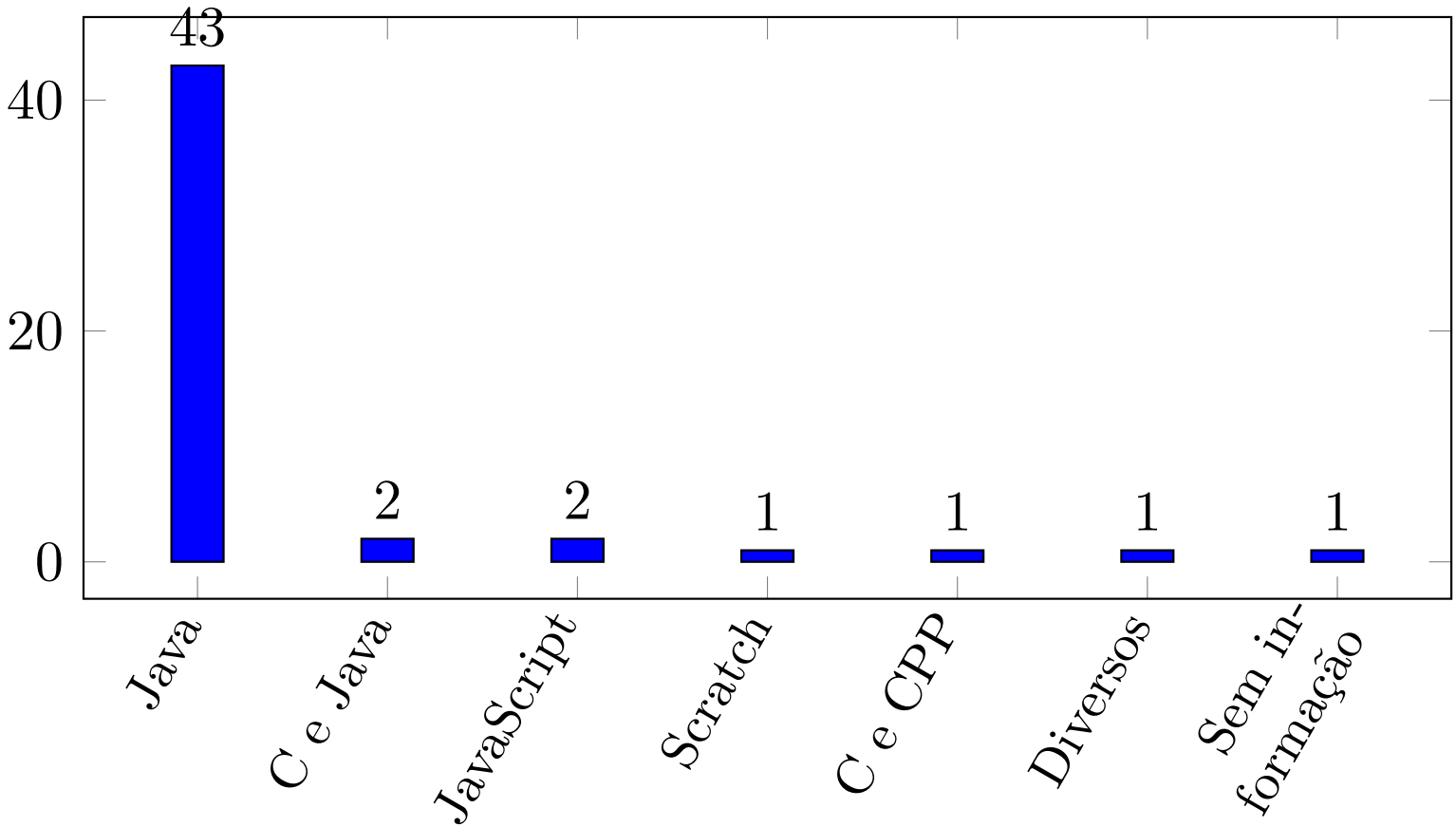
Revisão Sistemática da Literatura

QP1 - Quais oráculos de *bad smells* foram propostos na literatura?



Revisão Sistemática da Literatura

QP2 - Em quais linguagens de programação os sistemas que compõem os oráculos são implementados?



Revisão Sistemática da Literatura

QP3 - Qual é o tamanho dos sistemas que compõem os oráculos?

Programa	**Ocorrências**	**#classes**
Apache Xerces	15	736
Eclipse	8	1181-17167
Apache Ant	7	846
GanttProject	7	188-245
JFreeChart	7	86-775
ArgoUML	6	777-1415
JHotDraw	6	159-679
Apache Lucene	6	1762-2246
Apache Nutch	6	183-259
Apache Cassandra	5	305-826
jEdit	5	228-520

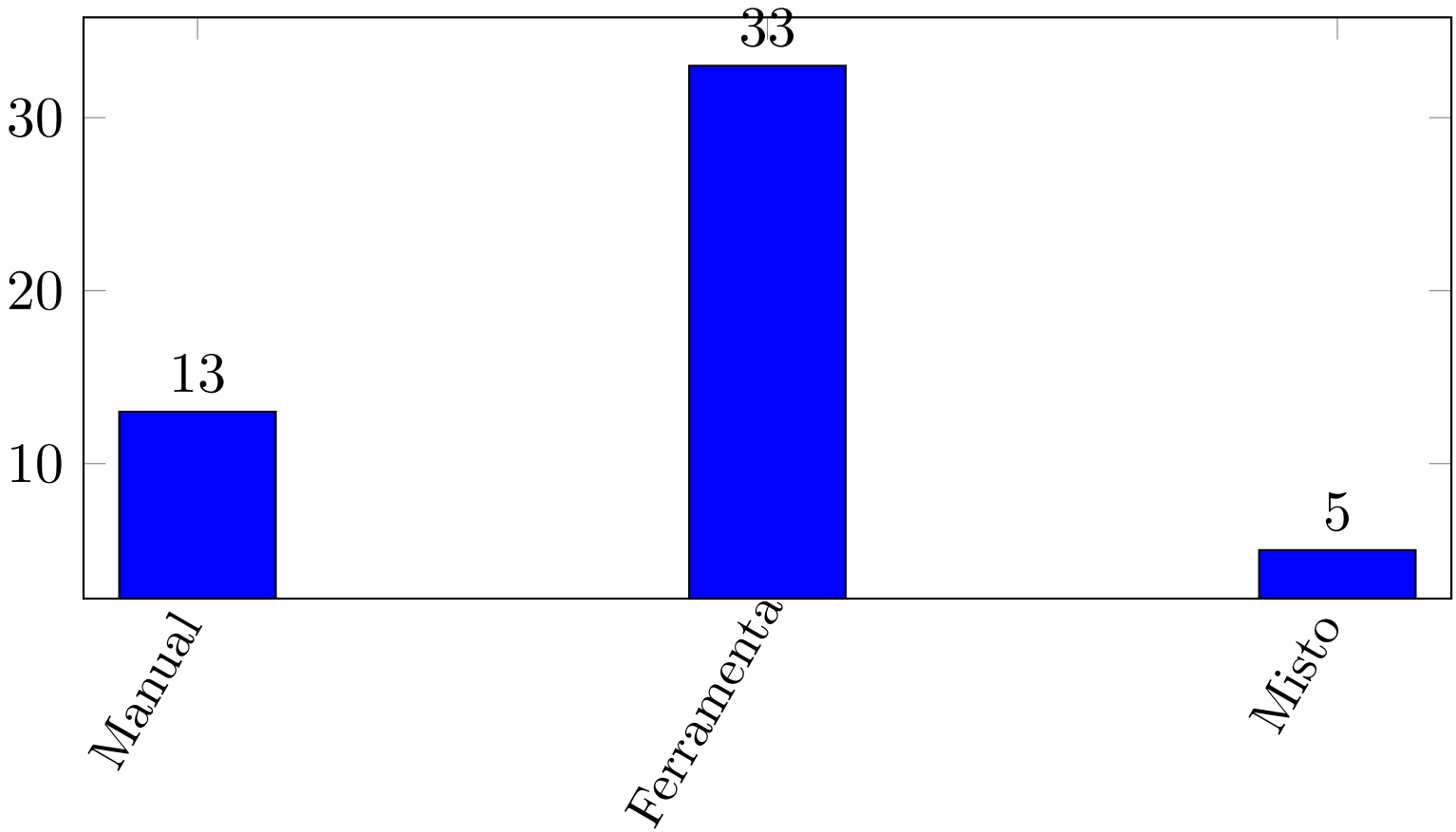
Revisão Sistemática da Literatura

QP4 - Quais *bad smells* são considerados pelos oráculos?

Bad Smell	**Número de artigos**
Blob ou *Large Class*	45
Long Method	26
Feature Envy	22
Data Class	17
Long Parameter List	15
ShotGun Surgery	15
Spaghetti Code	11
Duplicated Code	10
Functional Decomposition	10

Revisão Sistemática da Literatura

QP5 - Quais abordagens foram usadas para criar os oráculos?



Metodologia

Escolha dos Softwares

- Ser desenvolvido em Java
- Estar em repositório online
- Possuir relevância para a comunidade
- Possuir um tamanho relativamente considerável de classes e linhas para justificar o uso de uma ferramenta para a detecção
- Ser *open source*

Metodologia

Escolha dos Softwares

Softwares	***Snapshot***	**# de Classes**	**KLOC**
Aardvark	ff98d508	103	25
And engine	f25236e4	596	20
Apache Commons Codec	c6c8ae7a	103	23
Apache Commons Logging	d821ed3e	61	23
GanttProject	6d01a59	188	31

Metodologia

Escolha Bad Smells

Bad Smells	**Nível da ocorrência**
Blob ou *Large Class*	Classe
Long Method	Método
Feature Envy	Método
Data Class	Classe
Refused Bequest	Método

Metodologia

Plug-in `InspectBSmell`

O *plug-in* do Eclipse `InspectBSmell` foi desenvolvido para proporcionar as seguintes funcionalidades:

- assinalar a inspeção de uma classe
- cadastrar e excluir um *bad smell*
- cadastrar e excluir uma instância, ponto onde está localizado, de *bad smell*
- cadastrar, alterar e excluir de um comentário de uma instância de *bad smell*
- exportar oráculo em csv e json

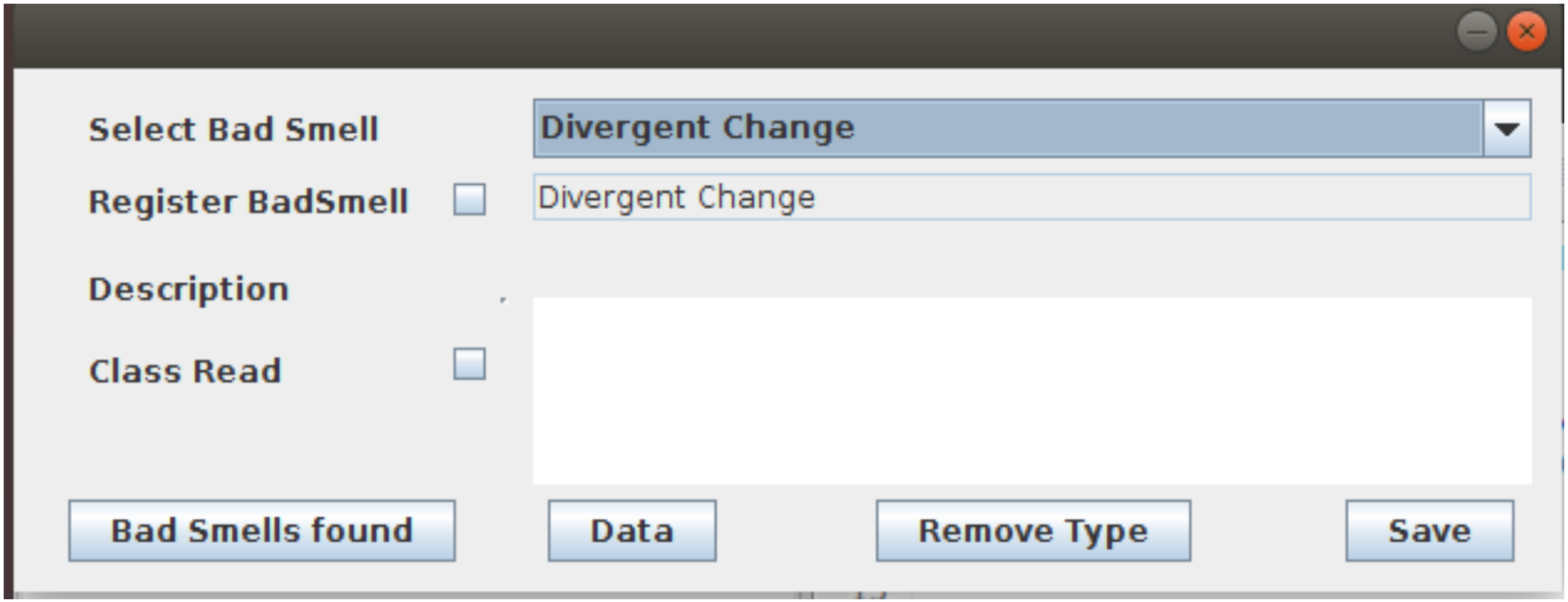
Metodologia

Inspeção pelo *Plug-in* `InspectBSmell`

1. Abra o projeto pelo Eclipse
2. Abra cada uma das classes que está em src
3. Procure por um dos seguintes *bad smell*: *god class*, *data class*, *long method*, *refused bequest*, *feature envy*
4. Se encontrar um *bad smell*, selecione o trecho de código que evidencia o *bad smell* e execute o comando Ctrl + 6, selecione o *bad smell* e coloque um comentário
5. Quando finalizar a inspeção da classe, execute o comando Ctrl + 6 e assinale a opção *Class Read* e clique em **Save**

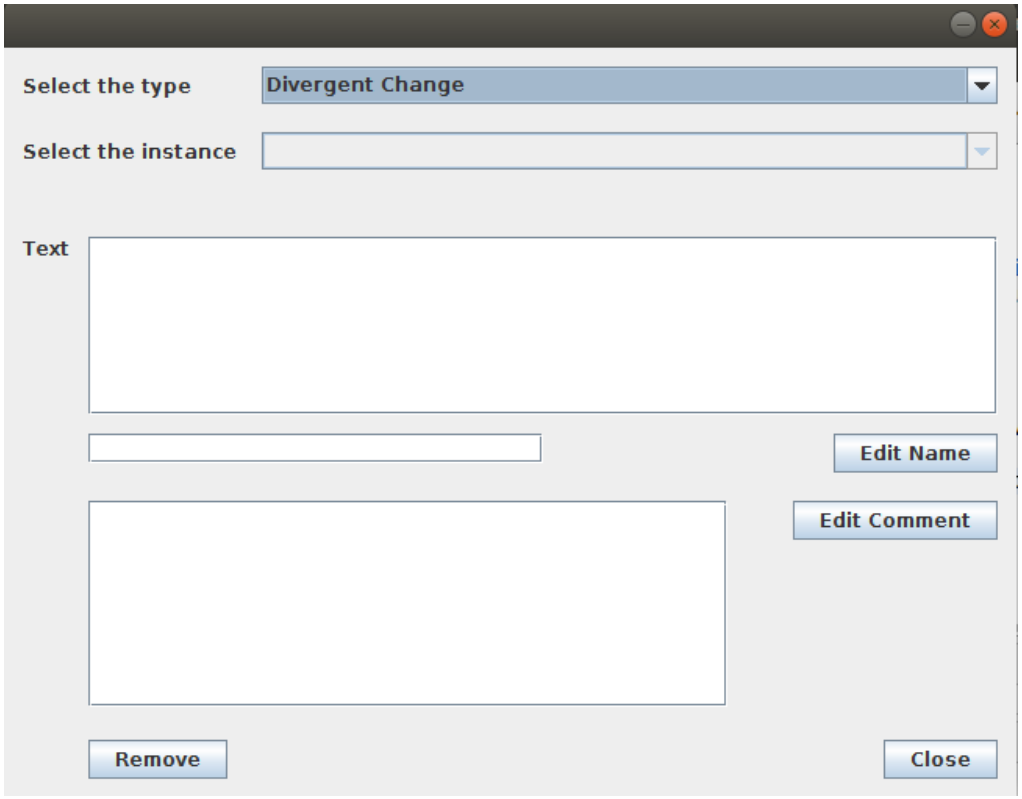
Metodologia

Plug-in: Cadastrar *Bad Smell*



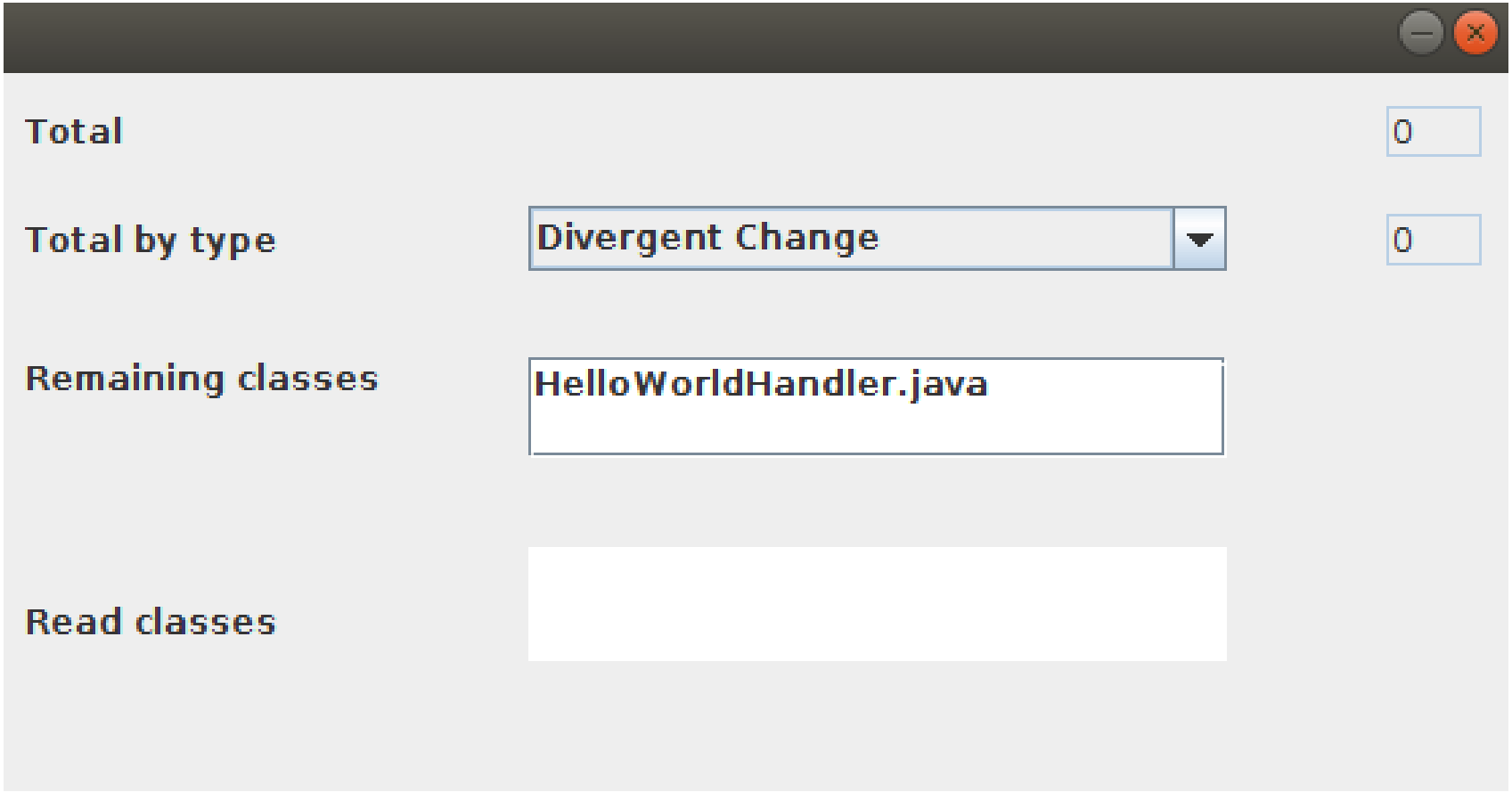
Metodologia

Plug-in: *Bad smells* encontrados



Metodologia

Plug-in: Resumo da Inspeção



Metodologia

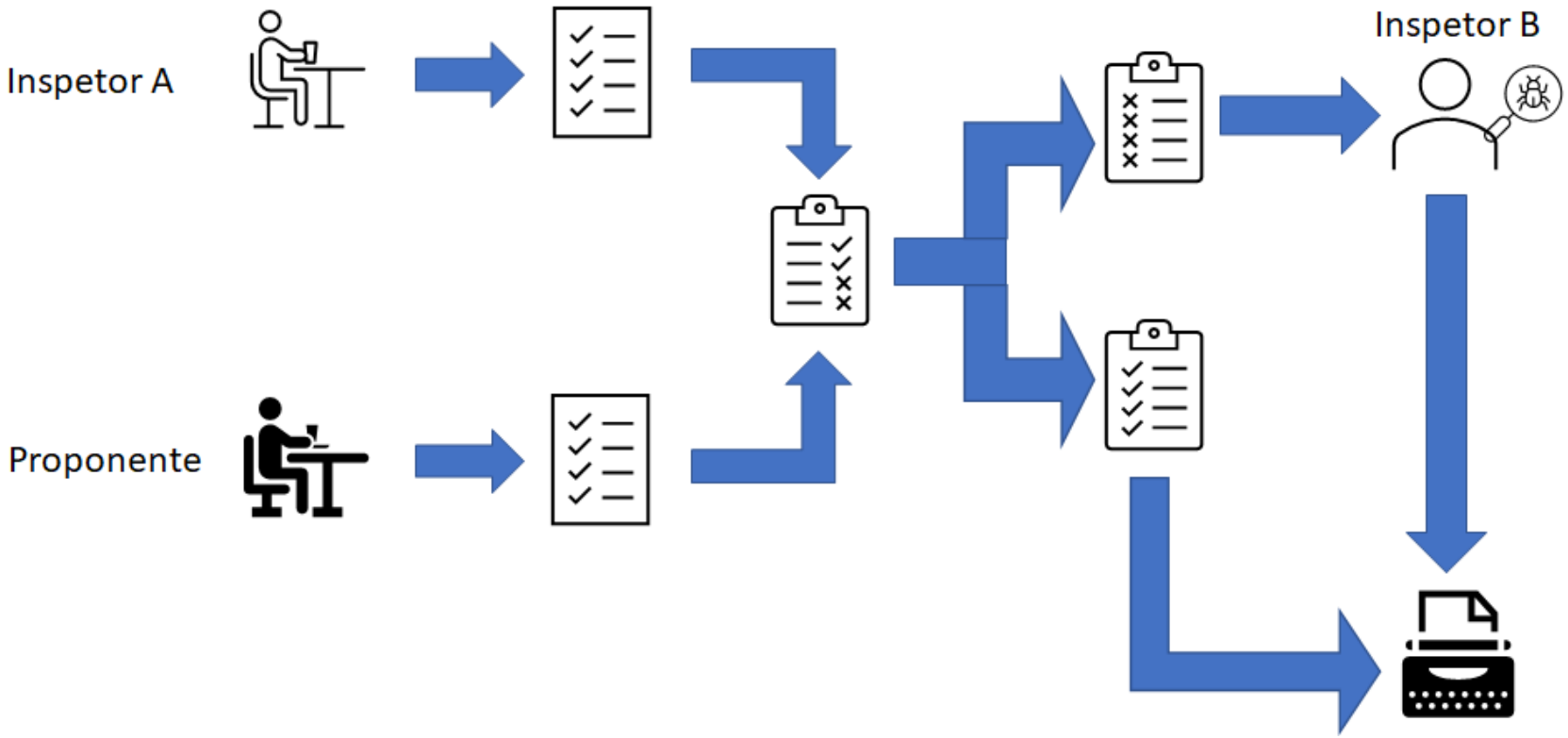
Consolidação Responsabilidades

Além da Inspeção do Autor, foram distribuídas as responsabilidades da segunda inspeção e da consolidação

Software	**Inspeção**	**Consolidação**
Aardvark	Colaborador 1	Colaborador 2
And engine	Colaborador 3	Colaborador 2
Apache Commons Codec	Colaborador 2	Colaborador 1
Apache Commons Logging	Colaborador 1	Colaborador 2
GanttProject	Colaborador 2	Colaborador 1

Metodologia

Consolidação



Resultados

Software	*LM*	*FE*	*DC*	*LC*	*RB*
Aardvark	0	0	4	6	1
Apache Commons Logging	0	0	1	2	0
GanttProject	12	13	18	48	0
And Engine	16	23	9	24	13
Apache Commons Codec	2	0	6	9	0

LM - *Long Method*
FE - *Feature Envy*
DC - *Data Class*
LC - *Blob* ou *Large Class*
RB - *Refused Bequest*

Resultados

Avaliação de Ferramentas de *Bad Smells*: Escolha das Ferramentas

1. Devem estar disponíveis para *download* gratuito pela *Internet*
2. Devem ser capazes de identificar *bad smells* em projetos Java
3. Devem:
   1. fornecer um ambiente próprio para a identificação e/ou coleta de dados ou
   2. ser possível adicioná-las ao ambiente `Eclipse` para serem executadas
4. Devem ser capazes de identificar os *bad smells* presentes na lista de *bad smells*

Resultados

Avaliação de Ferramentas de *Bad Smells*: Escolha das Ferramentas

Bad smell	**JSpIRIT**	**JDeodorant**	**FindSmells**
Data Class	Identifica	Não identifica	Identifica
Feature Envy	Identifica	Identifica	Identifica
Blob ou *Large Class*	Identifica	Identifica	Identifica
Long Method	Identifica*	Identifica	Identifica
Refused Bequest	Identifica	Não Identifica	Identifica

* *Brain Method* foi considerado sinônimo de *Long Method*

Resultados

Avaliação de Ferramentas de *Bad Smells*: Classificação

	Instância de *bad smell* identificada pela ferramenta	**Instância de *bad smell* não identificada pela ferramenta**
Instância de *bad smell* identificada pelo oráculo	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Instância de *bad smell* não identificada pelo oráculo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Resultados

	Ferramenta	*Precision*	*Recall*	*F-measure*
LM	JSpIRIT	0,67	0,55	0,60
FE	JDeodorant	0,20	0,80	0,32
DC	FindSmells	0,23	0,41	0,30
LC	JSpIRIT	0,51	0,72	0,60
RB	JSpIRIT	0,22	0,86	0,35

LM - *Long Method*
FE - *Feature Envy*
DC - *Data Class*
LC - *Blob* ou *Large Class*
RB - *Refused Bequest*

Conclusão

Contribuições

- Oráculo de *bad smells* de software para cinco sistemas e cinco *bad smells* disponível online
- A ferramenta `InspectBSmell` pode ser empregada para a criação de novos oráculos de *bad smells*
- Avaliação de três ferramentas de detecção de *bad smells* indica baixa acurácia, precisão e revocação
- Revisão Sistemática da Literatura identificou 12 oráculos de *bad smells* disponíveis online
- *Oracles of Bad Smells - a Systematic Literature Review* - artigo aceito para publicação na trilha *Research* do Simpósio Brasileiro de Engenharia de Software de 2020 (SBES 2020)

Agradecimentos

Fim!